AGL Media, Radio, and Telephony Bindings

Scott Murray
scott.murray@konsulko.com
About Konsulko Group

- Konsulko Group is a services company founded by embedded Linux veterans
- We do community and commercial embedded, Linux, and Open Source Software development
- Involved in AGL since 2015
- Have been working on demo application improvements for AGL since Fall 2016
- See www.konsulko.com for more information
Quick AGL Binding Overview

- The AGL application framework provides an API binding mechanism to allow abstracting an application’s UI from its back end logic
- This allows re-using application logic with different UI implementations (e.g. Qt and HTML5)
- More information
Binding Registration

- Binding implementation is a shared library
- A binding implementation:
  - Registers a unique binding name
  - Registers a list of binding verbs to perform actions
  - Exposes an initialization hook for the application framework
Application Binding Initialization

- Application packaging (widget) includes a config.xml file that:
  - Specifies the application type
  - Lists any bindings that the package requires
  - Lists any bindings that the package provides

- Application framework spawns an instance of afb-daemon
  - Loads and initializes the specified bindings
  - Runs the application, passing port number and authentication token arguments to it for binding access
  - Important to remember that each instance of the binding is separate

- More details
Application Binding Usage

- Submit requests in JSON format via HTTP or WebSocket
  - e.g. [ 2, "9999", "hvac/set", { "LeftTemperature" : 16} ]
- Receive request status (success or failure) and any additional requested data
- Responses are also in JSON format
- Can subscribe / unsubscribe for events
- Events arrive asynchronously via WebSocket
- More details
New bindings for Daring Dab

- Radio Binding
- Telephony Binding
- Media Binding
- Bluetooth / Settings Binding Enhancements
Radio Binding

● **Scope**
  ○ Enable AM/FM radio tuning
  ○ Expose station scanning interface

● **Initial Implementation**
  ○ Radio binding based on rtl-sdr SDR FM demodulation code previously used to build the QtMultimedia plugin from the Chinook release
  ○ Additional hooks added to FM demodulation code to add scanning support
  ○ Radio QML application reworked to use binding in place of QtMultimedia QRadio class
    ■ Only minimal changes were required, the QML interface for the binding emulates QRadio’s interface to a large degree
    ■ Application enhanced to add scanning support
  ○ Present in DD rc1
Radio Binding

- **Verbs**
  - **radio/frequency** - get/set frequency
    - Input: \{ "value" : INTEGER \}
      - Value is frequency in Hertz, e.g. 107100000
    - Input: None
    - Output: \{ "frequency" : INTEGER \}
  - **radio/band** - get/set band
    - Input: \{ "band" : STRING "<band>" \}
      - Band is “AM” or “FM”
    - Input: None
    - Output: \{ "band" : STRING "<band>" \}
Radio Binding

● Verbs (continued)
  ○ **radio/band_supported** - check band support
    ■ Input: { “band” : STRING “<band>” }
    ■ Output: { “supported” : BOOLEAN }
    ● Value is 0 or 1
  ○ **radio/frequency_range** - get band frequency range
    ■ Input: { “band” : STRING “<band>” }
    ■ Output: { "min" : INTEGER, "max" : INTEGER }
    ● Values are frequencies in Hertz
    ● e.g. { "min" : 87900000, "max" : 107900000 }
Radio Binding

- Verbs (continued)
  - **radio/frequency_step** - get band frequency step
    - Input: { "band" : STRING "<band>" }  
    - Output: { "step" : INTEGER }
    - Value is band frequency step in Hertz, e.g. 200000
  - **radio/start**
    - Input: None
    - Output: None
  - **radio/stop**
    - Input: None
    - Output: None
Radio Binding

- Verbs (continued)
  - radio/scan_start
    - Input: { "direction" : "forward" }
    - Input: { "direction" : "backward" }
    - Output: None
  - radio/scan_stop
    - Input: None
    - Output: None
  - radio/stereo_mode - get/set stereo mode
    - Input: { “value” : "mono"} 
    - Input: { “value” : "stereo"} 
    - Input: None 
    - Output: { “mode” : STRING "<mode>" }
Radio Binding

- Verbs (continued)
  - radio/subscribe
    - Input: { "value" : "frequency" }
    - Input: { "value" : "station_found" }
    - Output: None
  - radio/unsubscribe
    - Input: { "value" : "frequency" }
    - Input: { "value" : "station_found" }
    - Output: None
Radio Binding

● Events
  ○ `radio/frequency` - frequency has changed
    ■ { "value" : INTEGER }
  ○ `radio/station_found` - scanning has found a station
    ■ { "value" : INTEGER }
    ● Value is frequency of station
Radio Binding

● Future Development
  ○ Potentially add HD Radio support
    ■ There are some possible alternative hardware devices that do AM/FM/HD tuning
  ○ Metadata support (e.g. RDS)
Telephony Binding

● Scope
  ○ Manage telephony modems
  ○ Manage phone call lifecycle (dial, answer, hold, forwarding) operations

● Initial Implementation
  ○ Bluetooth Hands-Free Profile (HFP) device support only
  ○ Discover HFP capable devices
  ○ Originate a voice call
  ○ Answer an incoming voice call
  ○ Provide status and information on voice call connections
  ○ Phone application updated to use above binding functionality
  ○ Present in DD rc1
Telephony Binding

- Verbs
  - **telephony/dial** - dial a phone number
    - Input: `{ "number" : STRING "<phone number>" }`
    - Output: None
  - **telephony/hangup** - hangup an active phone call
    - Input: None
    - Output: None
  - **telephony/answer** - answer an incoming phone call
    - Input: None
    - Output: None
Telephony Binding

- Verbs (continued)
  - telephony/subscribe - subscribe to a telephony binding event
    - Input: \{ "value": "callStateChanged" \}
    - Input: \{ “value”: "incomingCall" \}
    - Input: \{ “value”: "dialingCall" \}
    - Input: \{ “value”: "terminatedCall" \}
    - Output: None
  - telephony/unsubscribe - unsubscribe from a telephony binding event
    - Input: \{ "value": "callStateChanged" \}
    - Input: \{ “value”: "incomingCall" \}
    - Input: \{ “value”: "dialingCall" \}
    - Input: \{ “value”: "terminatedCall" \}
    - Output: None
Telephony Binding

● Events
  ○ **telephony/callStateChanged** - state of a phone call has changed
    ■ { “state” : “active” } - call has been answered
    ■ { “state” : “held” } - call placed on hold
    ■ { “state” : “dialing” } - call is being dialed
    ■ { “state” : “alerting” } - call is alerting remote party (ringing remotely)
    ■ { “state” : “incoming” } - incoming call is ringing
    ■ { “state” : “waiting” } - incoming call is waiting due to active call
    ■ { "state" : “disconnected” } - call has been terminated
Telephony Binding

● Events (continued)
  ○ telephony/incomingCall - incoming call is ringing
    ■ { "clip" : "<incoming CLIP information>" } - numeric phone number information
  ○ telephony/dialingCall - outgoing call is being dialed
    ■ { "colp" : "<outgoing COLP information>" } - numeric phone number information
  ○ telephony/terminatedCall - call has been terminated
    ■ None
Telephony Binding

● Future Development
  ○ Merge incomingCall, dialingCall, and terminatedCall events into the callStateChanged event
  ○ In-call sending of dial tones (for conference bridges, etc.)
  ○ Manage access to multiple modems
  ○ Support SIM capable devices, including SIM specific operations (PIN handling, etc.)
  ○ Call waiting/hold/forwarding
  ○ Call volume support
  ○ CLIR support for dialing (restrict phone number)
Media Binding

● Scope
  ○ Enable control/detection of media from removable storage and Bluetooth
  ○ Provide media metadata
  ○ Media decode and output pipeline out of scope ATM, as strategic direction still being decided

● Initial Implementation
  ○ Media binding to report media insertion/removal
  ○ Media detection and path reporting
  ○ Receive metadata from Bluetooth binding
  ○ Access AVRCP Bluetooth binding media controls
  ○ MediaPlayer application updated to use above functionality
  ○ Functionality present in DD rc1, binding in rc2 (uploaded to Gerrit)
Media Binding

● Verbs
  ○ media/media_result - get all available multimedia
    ■ Input: None
    ■ Output: { “Media”: [ STRING “<file>”, … ] }  
      • e.g.: { “Media”: [ “/run/media/sda1/Track 1.ogg”, … ] }  
  ○ media/subscribe - subscribe to a media binding event
    ■ Input: { “value” : “media_added” }  
    ■ Input: { “value” : “media_removed” }  
    ■ Output: None  
  ○ media/unsubscribe - unsubscribe from a media binding event
    ■ Input: { “value” : “media_added” }  
    ■ Input: { “value” : “media_removed” }  
    ■ Output: None
Media Binding

● Events
  ○ media/media_added - media is attached to the device
    ■ { “Path”: STRING “<path>”, “Media”: [ STRING “<file>”, … ] }
    ● e.g.: { “Path” : “/run/media/sda1”, “Media”: [ “/run/media/sda1/Track 1.ogv”, … ] }
  ○ media/media_removed - media is removed from device
    ■ { “Path”: STRING “<path>” }
Media Binding Enhancements

- Future Development
  - Bluetooth binding integration in regards to AVRCP controls and stream metadata
  - Additional AVRCP controls (e.g. FastForward, Rewind, Volume Up/Down)
  - Allow switching between Bluetooth stream and local media
  - Return to Media application’s UI when Phone call ends if last window in focus
  - Allow local media playback to happen when Bluetooth A2DP connection is not streaming
Settings / Bluetooth Binding Enhancements

- **Scope**
  - Add extensions needed to expose required controls and metadata

- **Initial Implementation**
  - AVRCP Bluetooth binding controls
  - Media metadata, and position tracking
  - Coming in DD rc2
Settings / Bluetooth Binding

- Events
  - Bluetooth-Manager/device_updated - additional dictionary of metadata added
Settings / Bluetooth Binding Enhancements

● Future Development
  ○ Bluetooth binding needs to support inter-application access
  ○ MediaPlayer application updated to use Bluetooth binding
  ○ Phone application updated to use Bluetooth binding
  ○ Telephony call management removed from Bluetooth binding, replaced with standalone Telephony binding
Feedback

● These bindings are proofs of concept, oriented towards enabling the existing demo functionality
● Please suggest changes to enable your production use cases!
● Feedback channels:
  ○ IRC: #automotive on Freenode.net
  ○ Mailing list: https://lists.linuxfoundation.org/mailman/listinfo/automotive-discussions
  ○ Weekly developer call: https://wiki.automotivelinux.org/dev-call-info
  ○ JIRA: https://jira.automotivelinux.org
Resources

● Source git repositories
  ○ radio: git clone http://gerrit.automotivelinux.org/gerrit/apps/radio
  ○ phone: git clone http://gerrit.automotivelinux.org/gerrit/apps/phone
  ○ media: git clone http://gerrit.automotivelinux.org/gerrit/apps/mediaplayer
  ○ settings: git clone http://gerrit.automotivelinux.org/gerrit/apps/settings

● Binding Documentation
  ○ http://docs.automotivelinux.org/docs/apis_services/en/dev/

● Wiki
  ○ https://wiki.automotivelinux.org/start
Questions?
Glossary

- **API** Application Programming Interface
- **AVRCP** Audio/Video Remote Control Profile
- **CLIP** Calling Line Identification Presentation
- **CLIR** Calling Line Identification Restriction
- **COLP** Connected Line Identification Presentation
- **HFP** Hands-Free Profile
- **JSON** JavaScript Object Notation
- **RDS** Radio Descriptive Service
- **SDR** Software Defined Radio
- **SIM** Subscriber Identification Module